

Mathematics of Big Data, I

Lecture 5:

Naive Bayes, L1 Regularization, Scarsity, Lasso, Support Vector Machines, and Kernels

Weiqing Gu
Professor of Mathematics
Director of the Mathematics Clinic

Harvey Mudd College
Summer 2017

Today's topics

- Naive Bayes
- L1 Regularization, Sparsity, & Lasso
- Support Vector Machines
- Kernel Methods

Today's topics

- **Naive Bayes**
- L1 Regularization, Sparsity, & Lasso
- Support Vector Machines
- Kernel Methods

- Work out details with the students on the blackboard.

Recall the Chain Rule

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= \\ &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\ &= \dots \\ &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

Now the "naive" **conditional independence** assumptions come into play: assume that each feature F_i is conditionally **independent** of every other feature F_j for $j \neq i$, given the category C . This means that

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k).$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \cdots \\ &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k). \end{aligned}$$

Applications of Naïve Bayes

1. **Text categorization**: Judging a document belonging

- Legitimates,
- Sports,
- Politics,
- Sciences,
- Spams
- Etc.

- *Using word frequencies as features.*

2. **Medical diagnosis.**

More advanced method in above applications: **SVM**

Example on Naïve Bayes : Given historic weather Data and Yes-No Answers to Play Tennis

Attributes:

Class Attribute

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- $D = \{ (x^{(1)}, y_1), \dots, (x^{(k)}, y_k), \dots, (x^{(n)}, y_n) \}$
- Here $n = 14$
- Give $x^{(k)}$, the k^{th} day whether data.
- y belongs $\{1, 0\}$, 1=yes-play and 0=no-play,

Goal: Predict whether she is going to play today if knowing today's weather data

Outlook	Temp	Humidity	Windy	Play
Sunny	Cool	High	True	?

Give $X = (x_1 = \text{Sunny}, x_2 = \text{Cool}, x_3 = \text{high}, x_4 = \text{True})$
Predict for $Y=1$.

That is to find $P(Y=1 | X=\text{above})$

Weather Data Counts



Outlook	Temperature		Humidity		Windy		Play	
	Yes	No	Yes	No	Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4
Overcast	4	0	Mild	4	2	Normal	6	1
Rainy	3	2	Cool	3	1			
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5
Rainy	3/9	2/5	Cool	3/9	1/5			

We can use the table as a Model

Outlook	Temp	Humidity	Windy	Play
Sunny	Cool	High	True	?

Outlook	Temperature		Humidity		Windy		Play	
	Yes	No	Yes	No	Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4
Overcast	4	0	Mild	4	2	Normal	6	1
Rainy	3	2	Cool	3	1			
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5
Rainy	3/9	2/5	Cool	3/9	1/5			



Outlook	Temp	Humidity	Windy	Play
Sunny	Cool	High	True	?

For
Class=Yes

Outlook	Temperature	Humidity	Windy	Play
Yes	No	Yes	No	Yes
Sunny	2/9	3/5	Hot	2/5
Overcast	4/9	0/5	Mild	2/5
Rainy	3/9	2/5	Cool	1/5
		High	3/9	6/9
		Normal	1/5	3/9
			False	6/9
			True	2/5
				9/14
				5/14

Likelihood for the class play tennis equals to Yes
Yes = $2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

Likelihood of the New Day Outcome

Outlook	Temp	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes attribute Play can take

For each Class value (Yes and No)

$$\text{Yes} = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

$$\text{No} = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

Which class value is more likely?

Likelihood of the New Day Outcome

Convert into probabilities by normalization:

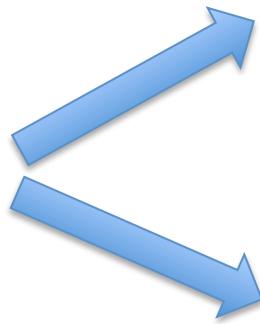
$$\text{Prob (Class = Yes)} = 0.0053 / (0.0053 + 0.0206) = 0.205$$

$$\text{Prob (Class = No)} = 0.0206 / (0.0053 + 0.0206) = 0.795$$

Probability for NOT
Play tennis is ~80%

What is “Naïve Bayes”?

Naïve Bayes



Naive Bayes Models

Naive Bayes classifiers

What is “Naïve Bayes Models”?

- **Naive Bayes Model is a conditional probability model:**
- Given a problem instance to be classified, represented by a vector

$$\mathbf{x} = (x_1, \dots, x_n)$$

representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

for each of K possible outcomes or classes C_k

Recall Bayes' theorem, the conditional probability

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

Bayesian probability terminology:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

What are Probabilistic Classifiers?

Probabilistic classifier is a classifier that is able to predict, given a sample input, a probability distribution over a set of classes, rather than only outputting the most likely class that the sample should belong to. Probabilistic classifiers provide classification with a degree of certainty, which can be useful in its own right, or when combining classifiers into ensembles.

Binary probabilistic classifiers are also called binomial regression models in statistics. In econometrics, probabilistic classification in general is called discrete choice.

"Hard" classification: using the optimal decision rule:

$$\hat{y} = \arg \max_y \Pr(Y = y | X)$$

That is to say: the predicted class is that which has the highest probability.

What are naive Bayes classifiers?

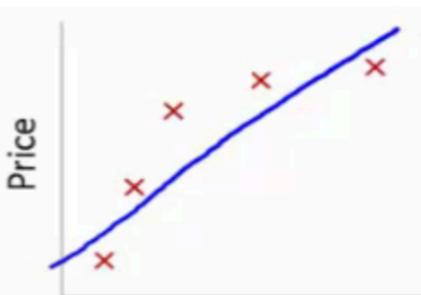
- Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Today's topics

- Naive Bayes
- L1 Regularization, Sparsity, & Lasso
- Support Vector Machines
- Kernel Methods

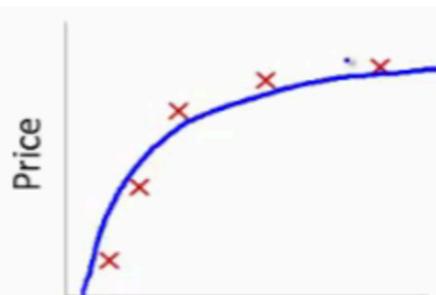
Regularization and Model Selection

- Regularization is a technique used to solve the overfitting problem in statistical models.
- What is overfitting?



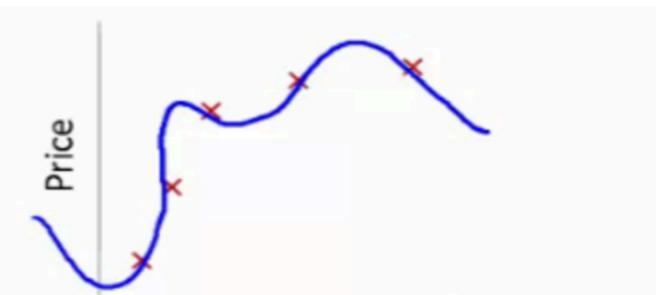
$$\theta_0 + \theta_1 x$$

High bias
(underfit)



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

"Just right"



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

High variance
(overfit)

Example: When someone wants to predict the price of a house based on its size, she tries to select among several different models. For instance, she tries to use polynomial regression model $h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_k x^k)$, and wish to decide if which k would be best fit the data.

Q: How can we automatically select a model that avoid high bias and high variance?

What is regularization in machine learning?

When do we need to use it?

- Regularization is a technique used in an attempt to solve the overfitting problem in statistical models.
- Here is an intuitive example: Let's say we want to model and predict the wage of someone based on his age.
- We will first try a linear regression model with age as an independent variable and wage as a dependent one. This model will mostly fail, since it is too simple.

- Then, you may add more data into your model such as *the sex and the education of each individual*.
- Your model becomes **more interesting and more complex**.
- You find out that your results are quite good but not as perfect as you wish.
- So you add more variables: location, profession of parents, social background, number of children, weight, number of books, preferred color, best meal, last holidays destination and so on and so forth
- But then **your model becomes overfitting**.
- How could we teach the machine select the most important features/variables?

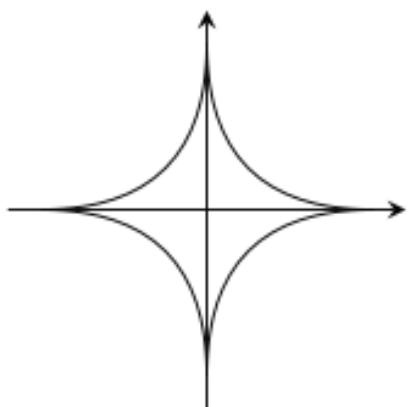
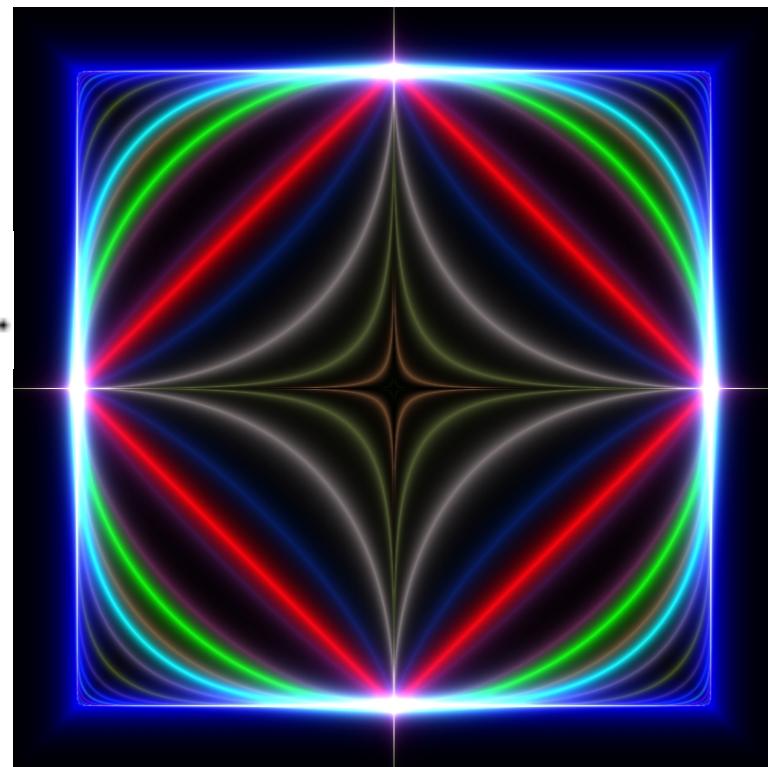
Key: Regularization is to teach machine to find a model avoid under- or over-fitting, do just “right” fitting.

We will need certain metric so that the machine can use it to set non-important variables to zeros! We consider L_1 metric.

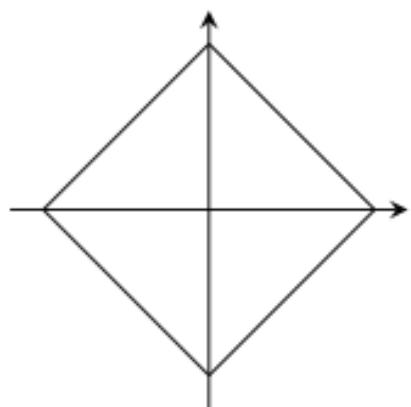
What is L_p norm?

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}.$$

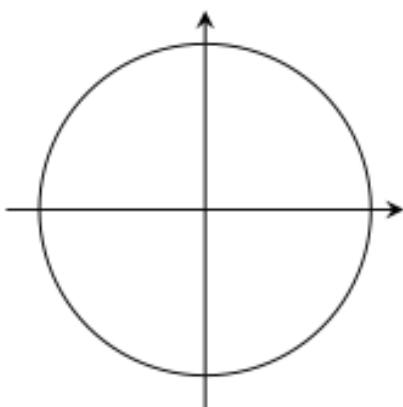
$$\|x\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$



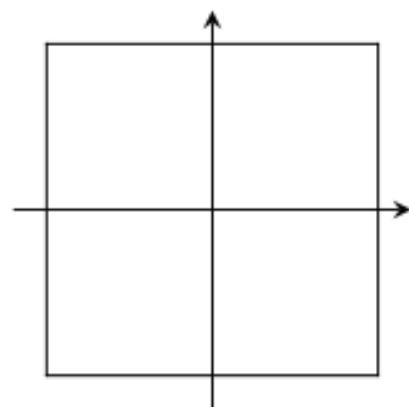
$$p = \frac{1}{2}$$



$$p = 1$$



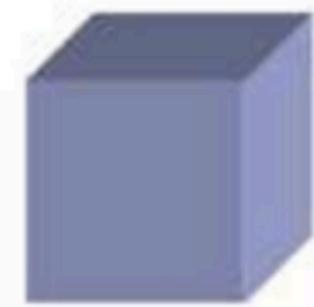
$$p = 2$$



$$p = \infty$$

L_p Norm For vectors in \mathbb{R}^3

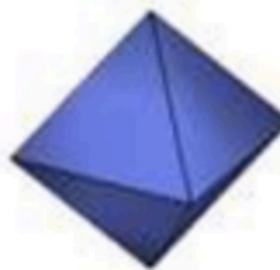
$$\|\mathbf{x}\|_p \leq 1$$



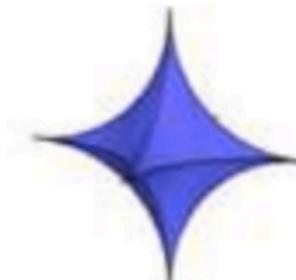
$$p = \infty$$



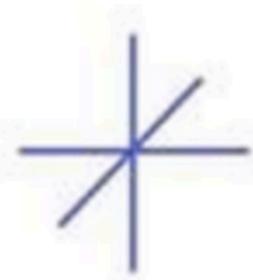
$$p = 2$$



$$p = 1$$



$$0 < p < 1$$



$$p = 0$$

Note: As the value of p decreases, the size of the corresponding L_p space also decreases. This can be seen visually when comparing the the size of the spaces of signals, in three dimensions, for which the L_p norm is less than or equal to one. The volume of these L_p “balls” decreases with p .

What is LASSO?

See the math on the next slide first.

LASSO= Least Absolute Shrinkage and Selection Operator

- The LASSO is a regression method that involves penalizing the absolute size of the regression coefficients.
- By penalizing (or equivalently constraining the sum of the absolute values of the estimates) you end up in a situation where some of the parameter estimates may be exactly zero. The larger the penalty applied, the further estimates are shrunk towards zero.
- This is convenient when we want some automatic feature/variable selection, or when dealing with highly correlated predictors, where standard regression will usually have regression coefficients that are 'too large or too many'.

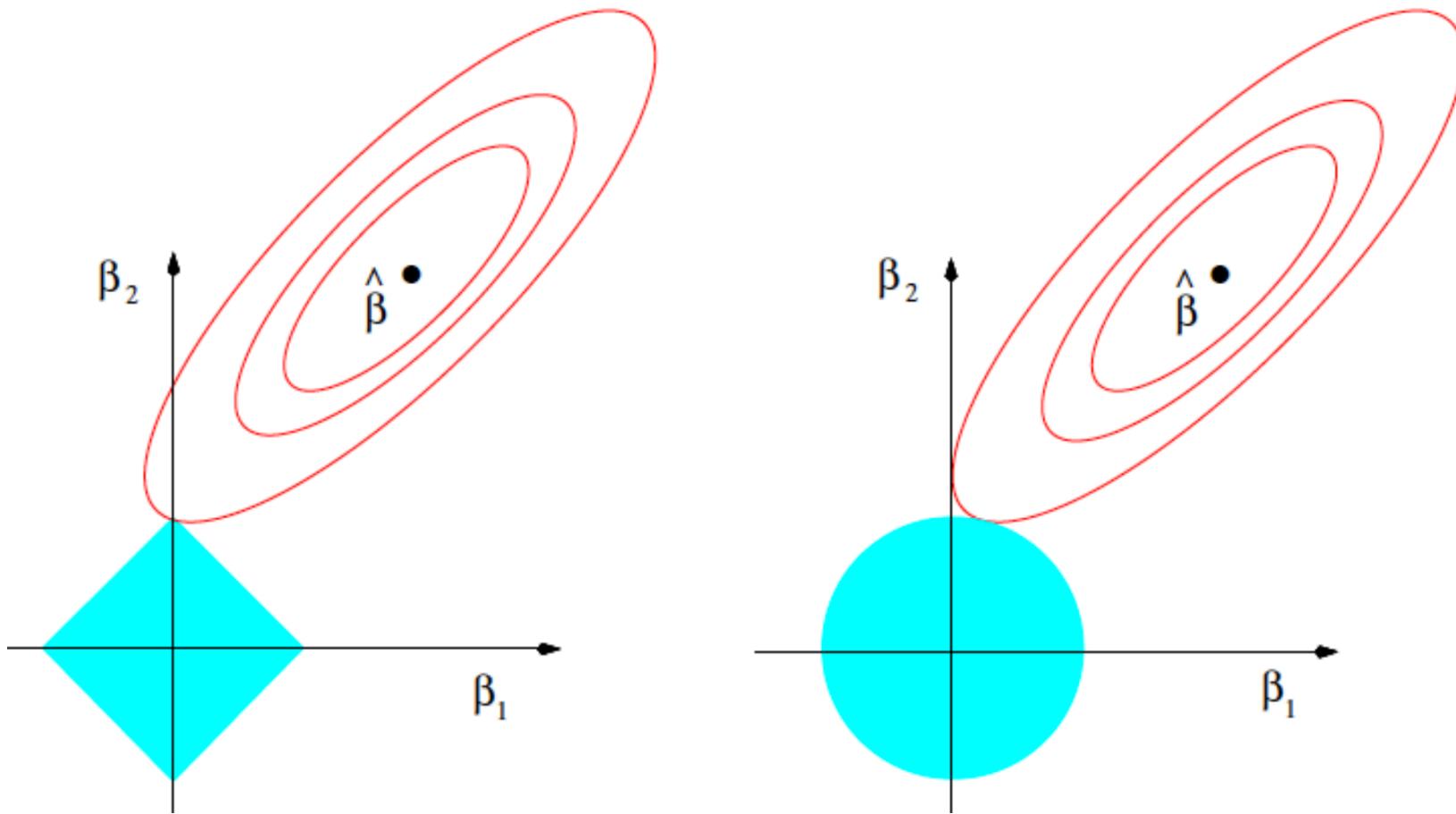
Compare LASSO and Ridge Regressions

$$\begin{aligned}\hat{\beta}^{\text{lasso}} &= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \|y - X\beta\|_2^2 + \lambda \sum_{j=1}^p |\beta_j| \\ &= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}\end{aligned}$$

$$\hat{\beta}^{\text{ridge}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$$

The only difference between the lasso problem and ridge regression is that the latter uses a (squared) ℓ_2 penalty $\|\beta\|_2^2$, while the former uses an ℓ_1 penalty $\|\beta\|_1$. But even though these problems look similar, their solutions behave very differently

Why does the lasso give sparse solution (i.e. many zero coefficients)?



(From page 71 of ESL)

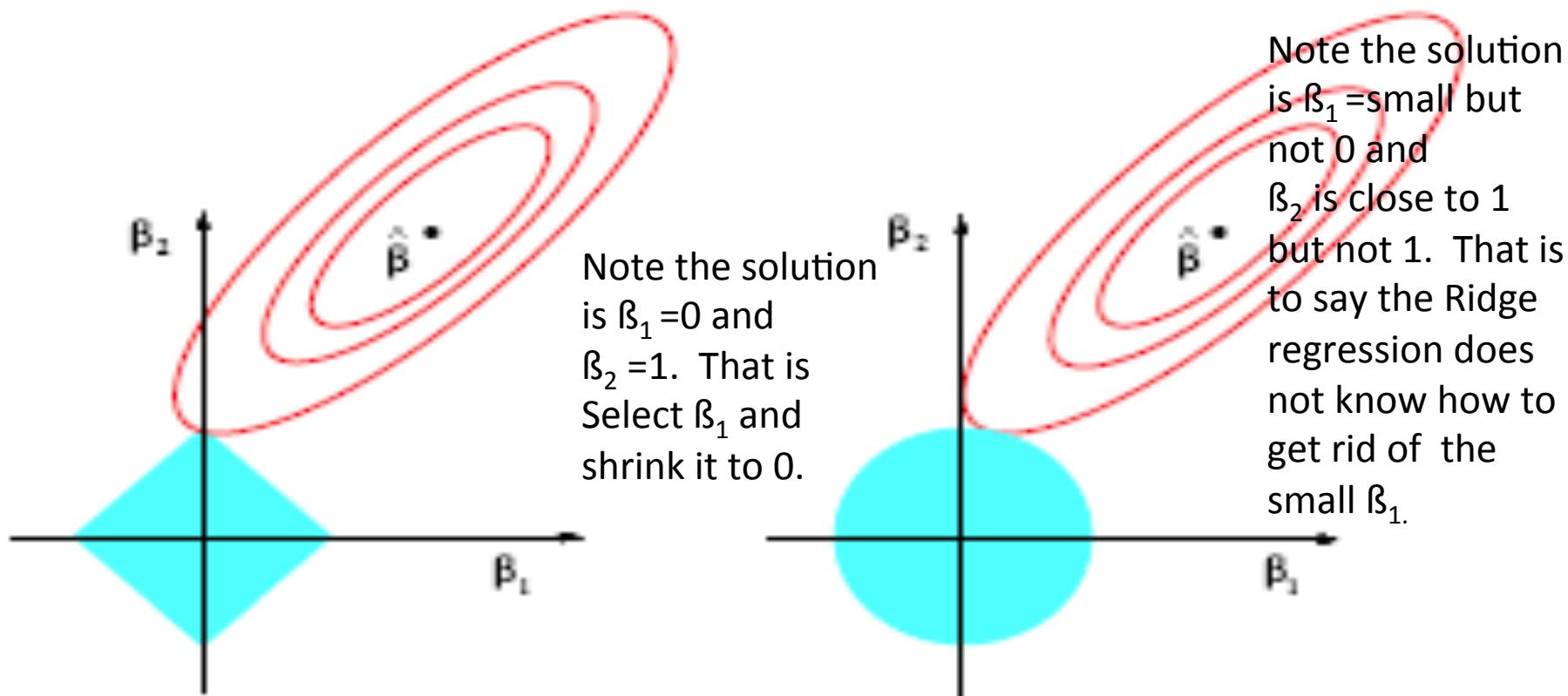


Figure 3.12: *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

Key: Lesson is able to perform variable selection in the linear model.

The tuning parameter λ controls the strength of the penalty, and (like ridge regression) we get $\hat{\beta}^{\text{lasso}} = \text{the linear regression estimate when } \lambda = 0,$ and $\hat{\beta}^{\text{lasso}} = 0 \text{ when } \lambda = \infty$

For λ in between these two extremes, we are balancing two ideas: fitting a linear model of y on X , and shrinking the coefficients. But the nature of the ' 1 ' penalty causes some coefficients to be shrunken to zero exactly.

This is what makes the lasso substantially different from ridge regression: it is able to perform variable selection in the linear model.

As λ increases, more coefficients are set to zero (less variables are selected), and among the nonzero coefficients, more shrinkage is employed.

Let the machine do the job for you!

- 1. Randomly split S into Strain (say, 70% of the data) and S_{cv} (the remaining 30%). Here, S_{cv} is called the hold-out cross validation set.
- 2. Train each model M_i on Strain only, to get some hypothesis h_i .
- 3. Select and output the hypothesis h_i that had the smallest error $\epsilon_{S_{cv}}(h_i)$ on the hold out cross validation set.

Regularization path

As we increase λ , the solution vector $\hat{\mathbf{w}}(\lambda)$ will tend to get sparser, although not necessarily monotonically. We can plot the values $\hat{w}_j(\lambda)$ vs λ for each feature j ; this is known as the **regularization path**.

This is illustrated for ridge regression in Figure 13.7(a), where we plot $\hat{w}_j(\lambda)$ as the regularizer λ decreases. We see that when $\lambda = \infty$, all the coefficients are zero. But for any finite value of λ , all coefficients are non-zero; furthermore, they increase in magnitude as λ is decreased.

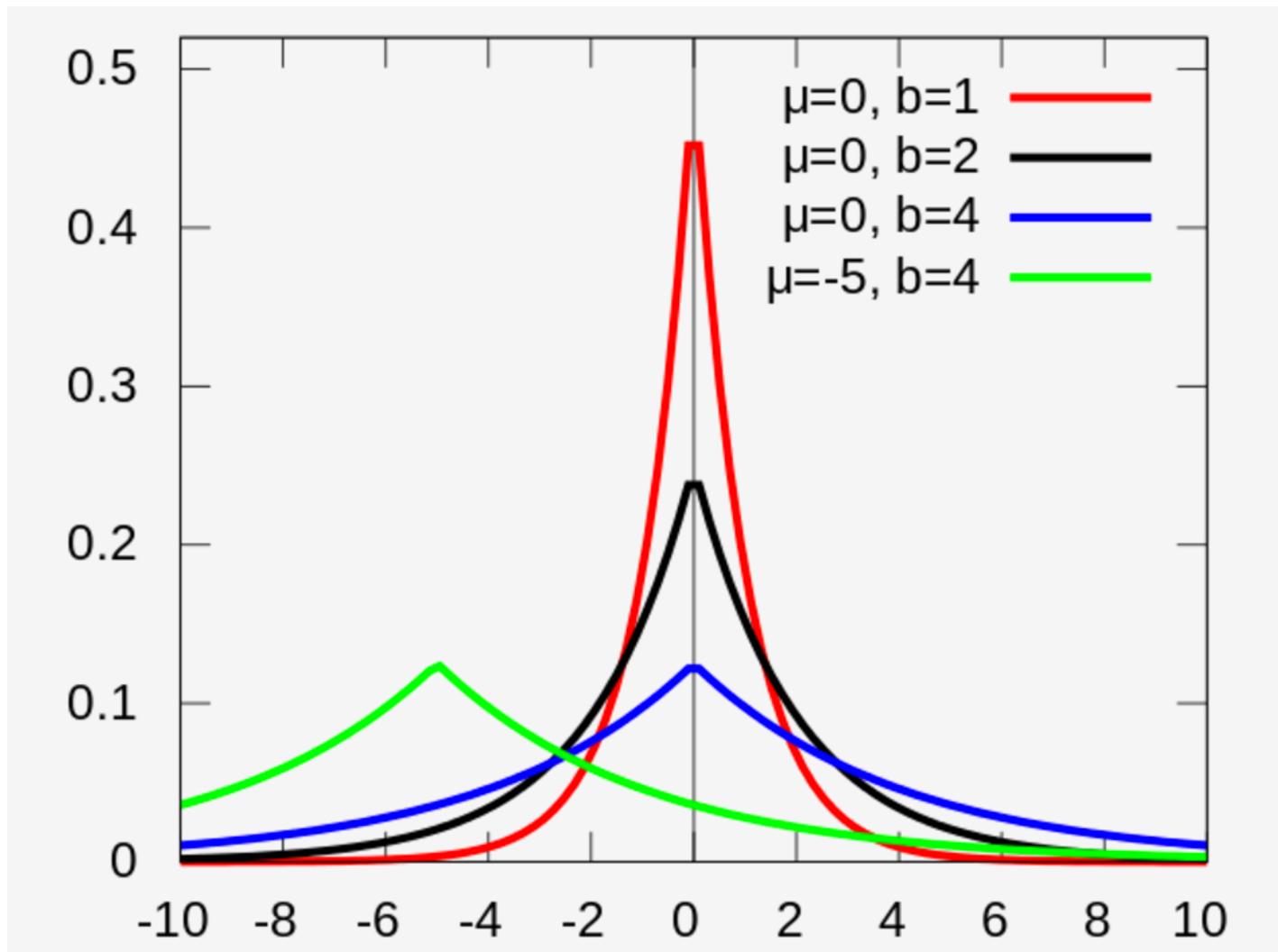
In Figure 13.7(b), we plot the analogous result for lasso. As we move to the right, the upper bound on the ℓ_1 penalty, B , increases. When $B = 0$, all the coefficients are zero. As we increase B , the coefficients gradually “turn on”. But for any value between 0 and $B_{max} = \|\hat{\mathbf{w}}_{OLS}\|_1$, the solution is sparse.⁴

Laplace distribution

A random variable has a Laplace(μ , b) distribution if its probability density function is
 L_1 norm

$$f(x | \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right)$$
$$= \frac{1}{2b} \begin{cases} \exp\left(-\frac{\mu-x}{b}\right) & \text{if } x < \mu \\ \exp\left(-\frac{x-\mu}{b}\right) & \text{if } x \geq \mu \end{cases}$$

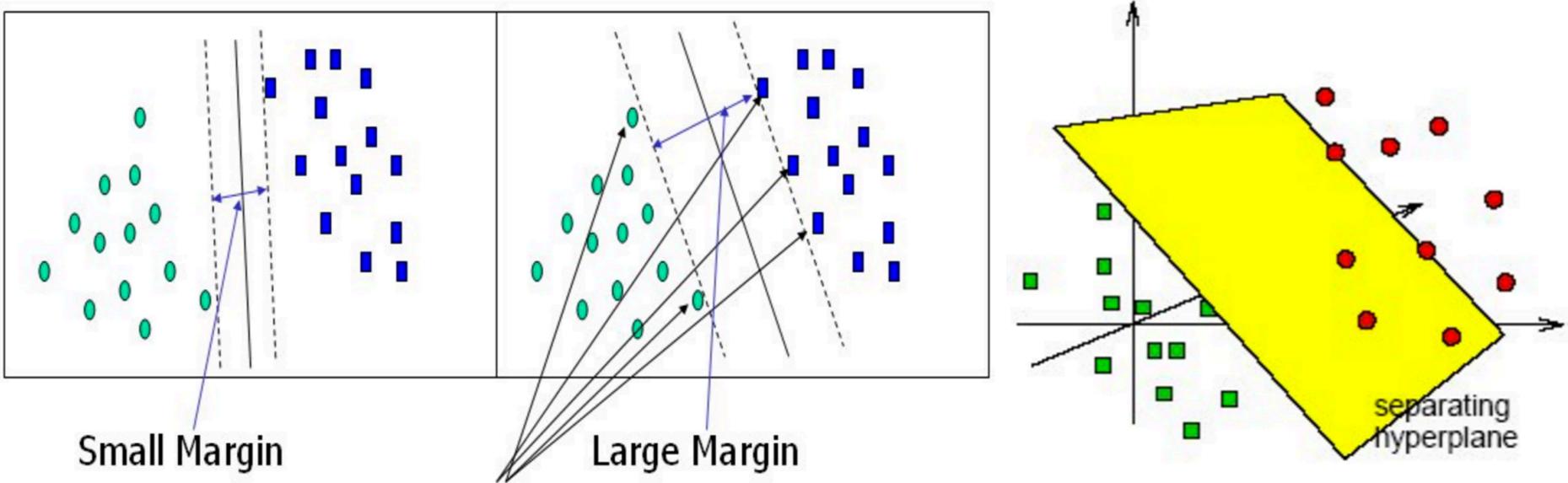
Examples of Laplace distribution



Today's topics

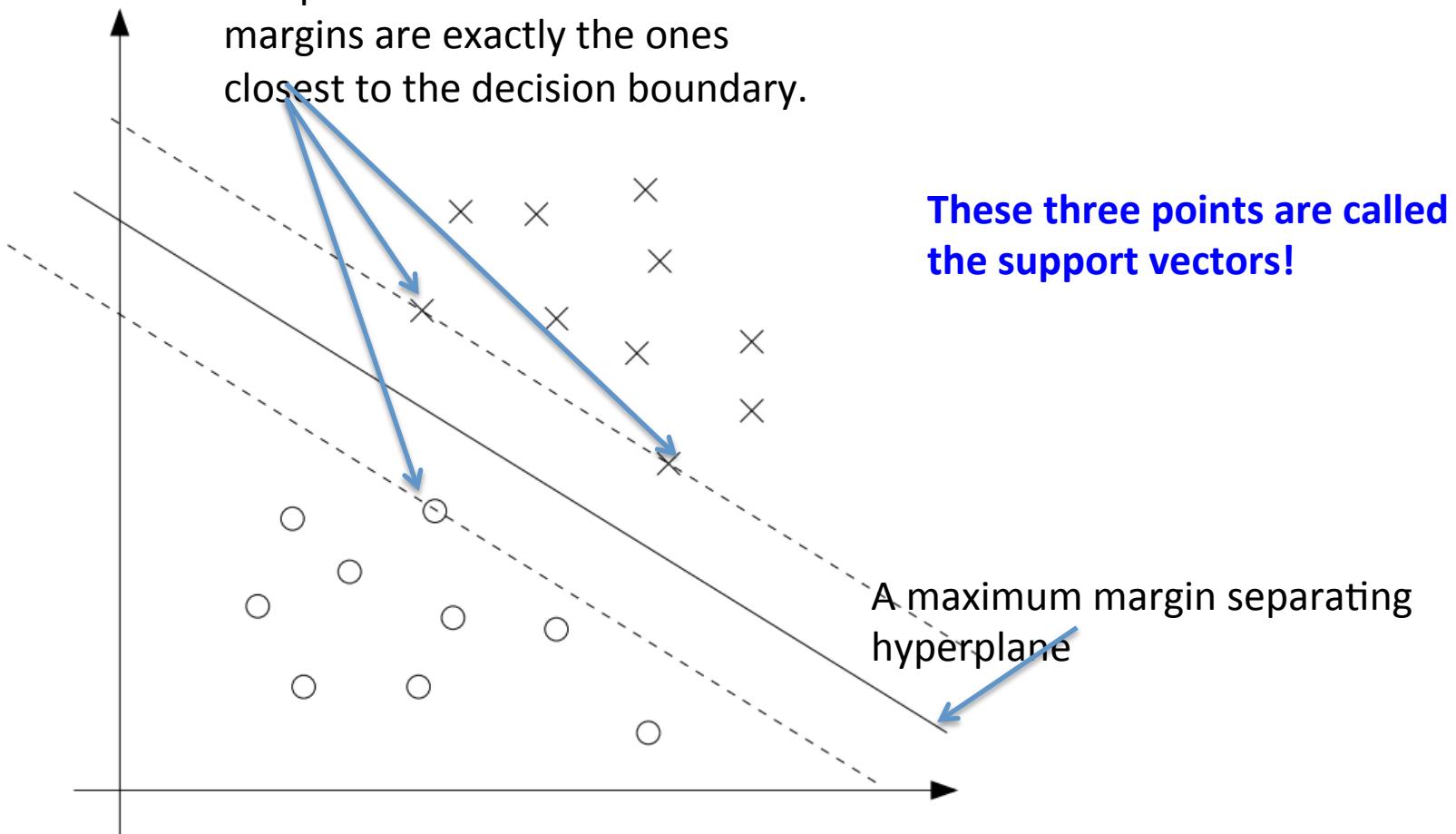
- Naive Bayes
- L1 Regularization, Sparsity, & Lasso
- **Support Vector Machines (SVM)**
- Kernel Methods

Support Vector Machine (SVM)



- Key: Gain an geometric intuition:
- ***Want to maximize the margin to increase the confidence of your prediction.***
- ***SVM is a typical example of Machine learning from of Geometric perspective.***
- There is a procedure for geometric approaches.

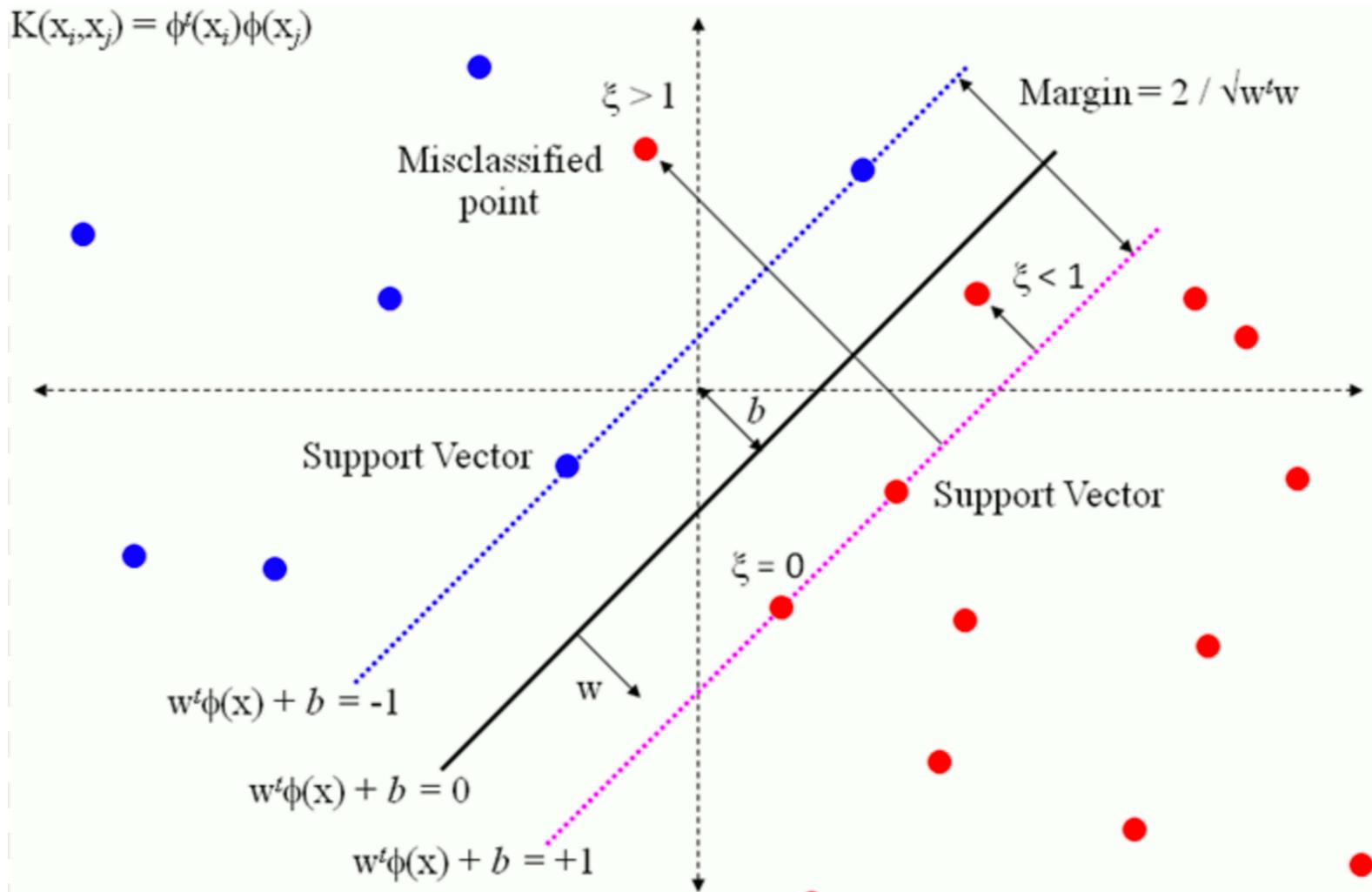
Separating Hyper-plane and Support Vectors



Geometric Analysis behind the Support Vector Machine (SVM)

- One of the powerful machine learning techniques.
- Working out details with the students on the board.

Here we first assume that the data is linear separable. We will deal with the case when data is not linear separable **later**; just need to make a clever “transformation”.



The following slides closely follow Prof. Andrew Ng' lecture notes on SVM

Functional Margin

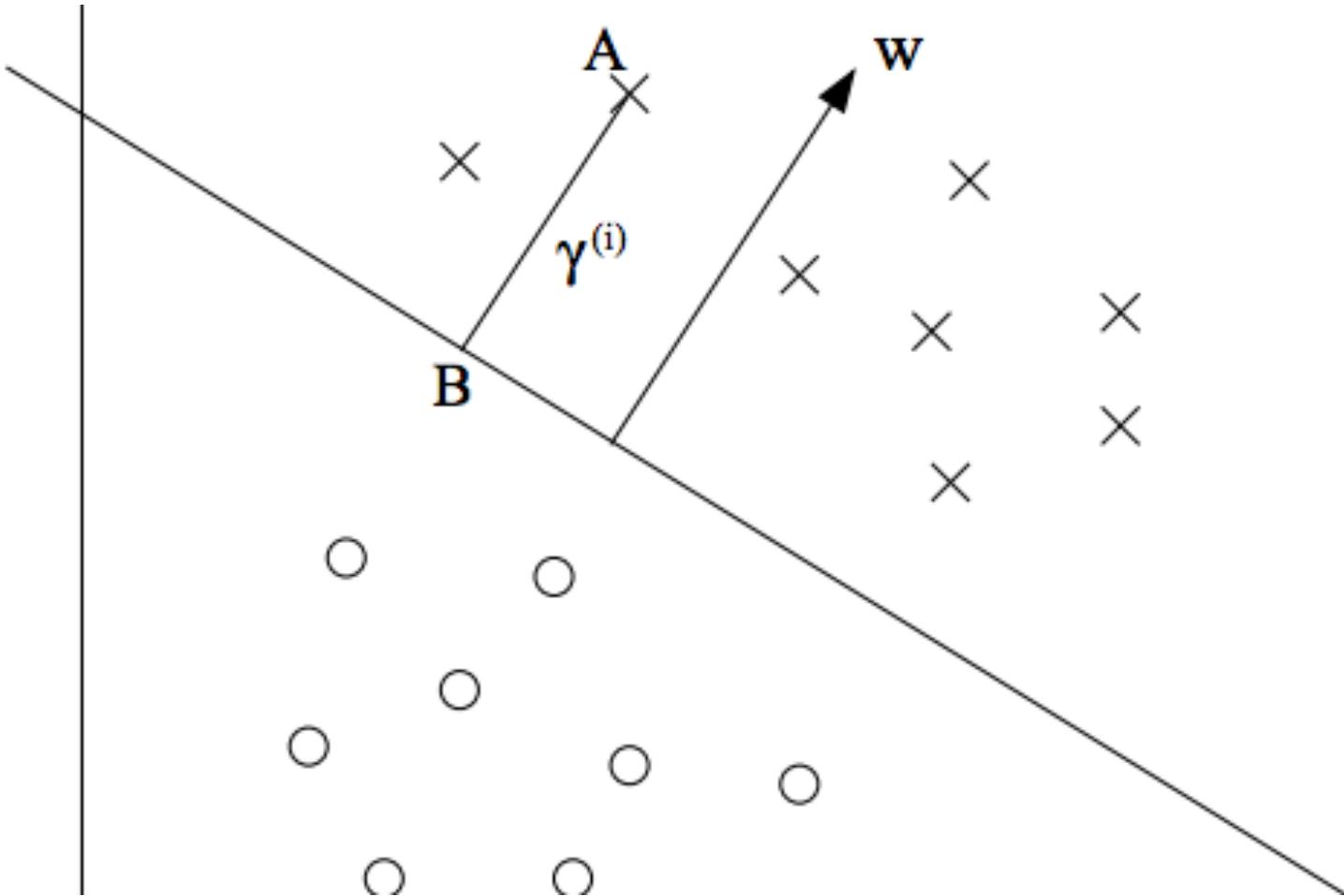
- Recall: we define the functional margin of (w, b) with respect to the training example

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b).$$

Given a training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \dots, m\}$, we also define the function margin of (w, b) with respect to S to be the smallest of the functional margins of the individual training examples. Denoted by $\hat{\gamma}$, this can therefore be written:

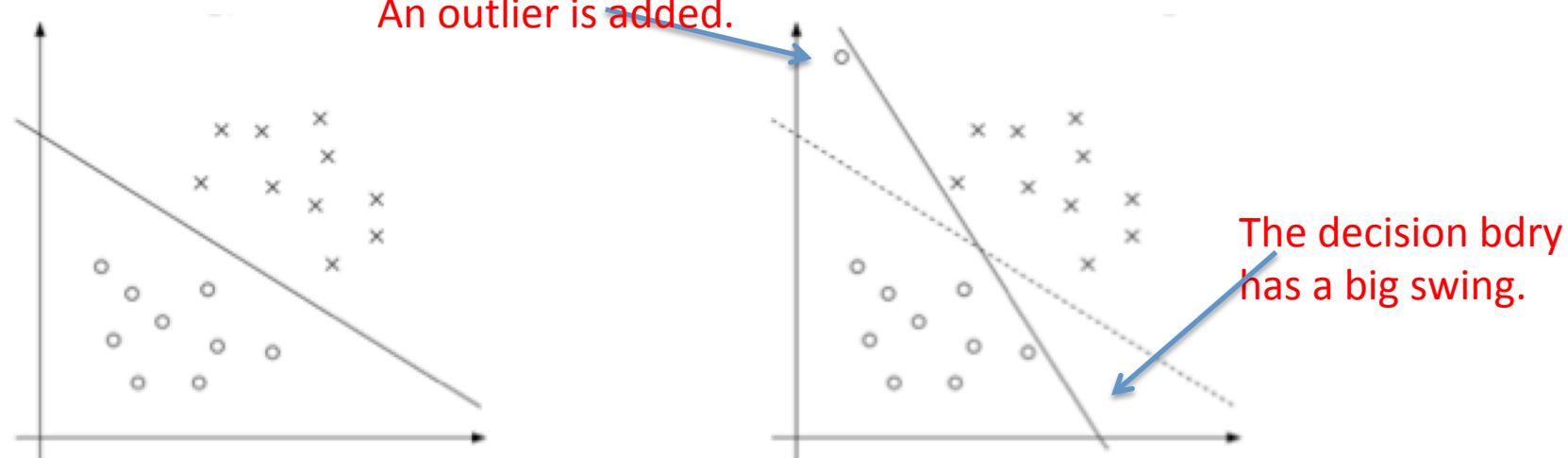
$$\hat{\gamma} = \min_{i=1, \dots, m} \hat{\gamma}^{(i)}.$$

Geometric Margins



Q: How to Teach the Machine to Deal w/ Margin Swinging Problem?

The figure shows an optimal margin classifier, and when a single outlier is added, it causes the decision boundary to make a dramatic swing, and the resulting classifier has a much smaller margin.



Ans: Using L₁ regularization!

$$\begin{aligned} \min_{\gamma, w, b} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \dots, m \\ & \xi_i \geq 0, \quad i = 1, \dots, m. \end{aligned}$$

The parameter C controls the relative weighting between the twin goals of making the $\|w\|^2$ small (which we saw earlier makes the margin large) and of ensuring that most examples have large functional margin.

Key Steps in the Geometric Approach

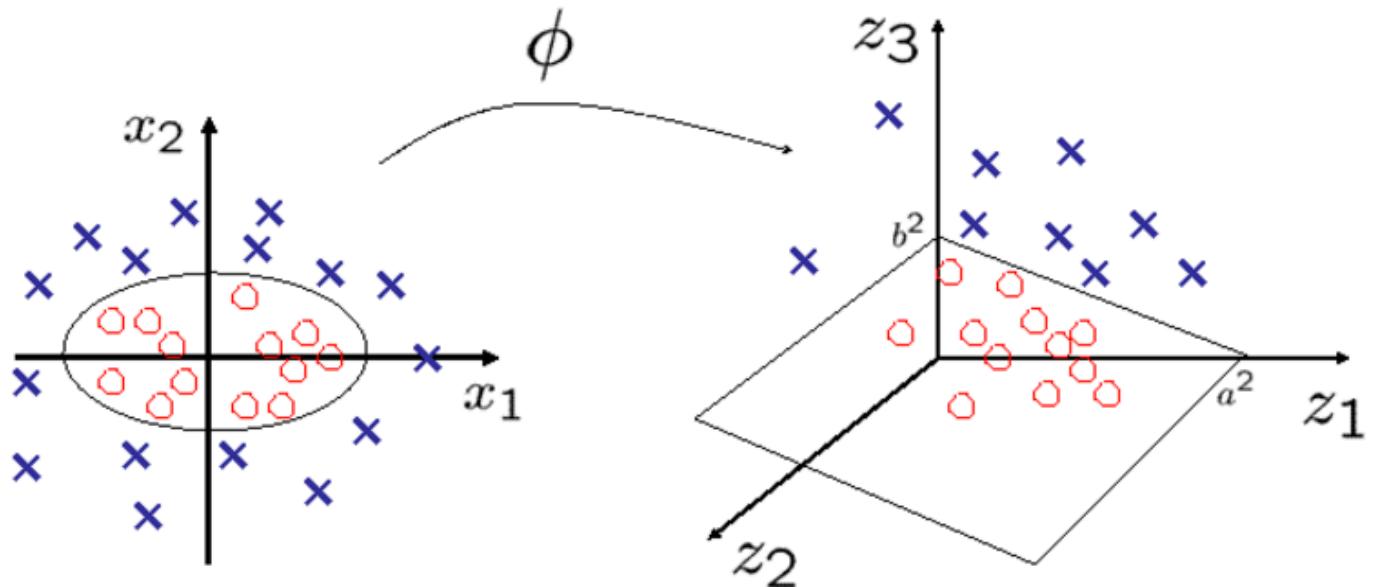
- Gain an geometric intuition: For SVM--Want to maximize the margin to increase the confidence of your prediction.
- Write down your geometrical intuition Mathematically.
- Come up with a cost function: Here is the margin. Want to maximize the margin.
- Figure out what are the constraints involved.
- Make the problem into a convex problem (if possible to do so) if the optimization you formed is not convex.
- For SVM, we will make it into a quadratic program problem.
- Use exiting QP software package to solve the problem.
- Use different way to solve it. (Later use an dual method.)
- Generate to solve more complicated problems/cases.

Today's topics

- Naive Bayes
- L1 Regularization, Sparsity, & Lasso
- Support Vector Machines
- **Kernel Methods**

Kernel Methods

- Feature map
- Kernel



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

This is a plane!
When $z_1 = 0, z_3 = b^2$,
and
When $z_3 = 0, z_1 = a^2$.
And z_2 and by any
thing in \mathbb{R} .

Feature Map

- Examples

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}.$$

Example

Suppose $x, z \in \mathbb{R}^n$, and consider

$$K(x, z) = (x^T z)^2.$$

$$\begin{aligned} K(x, z) &= \left(\sum_{i=1}^n x_i z_i \right) \left(\sum_{j=1}^n x_j z_j \right) \\ &= \sum_{i=1}^n \sum_{j=1}^n x_i x_j z_i z_j \\ &= \sum_{i,j=1}^n (x_i x_j)(z_i z_j) \end{aligned}$$

Thus, we see that $K(x, z) = \phi(x)^T \phi(z)$, where the feature mapping ϕ is given (shown here for the case of $n = 3$) by

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}.$$

Note that whereas calculating the high-dimensional $\phi(x)$ requires $O(n^2)$ time, finding $K(x, z)$ takes only $O(n)$ time—linear in the dimension of the input attributes.

$$K(x, z) = (x^T z + c)^2$$

$$= \sum_{i,j=1}^n (x_i x_j)(z_i z_j) + \sum_{i=1}^n (\sqrt{2c} x_i)(\sqrt{2c} z_i)$$

$$\phi(x) =$$

$$K(x, z) = \phi(x)^T \phi(z)$$

$$\begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \\ \sqrt{2c} x_1 \\ \sqrt{2c} x_2 \\ \sqrt{2c} x_3 \\ c \end{bmatrix}$$

More broadly, the kernel $K(x, z) = (x^T z + c)^d$ corresponds to a feature mapping to an $\binom{n+d}{d}$ feature space, corresponding of all monomials of the form $x_{i_1} x_{i_2} \dots x_{i_k}$ that are up to order d . However, despite working in this $O(n^d)$ -dimensional space, computing $K(x, z)$ still takes only $O(n)$ time, and hence we never need to explicitly represent feature vectors in this very high dimensional feature space.

Since the algorithm can be written entirely in terms of the inner products $\langle x, z \rangle$, this means that we would replace all those inner products with $\langle \phi(x), \phi(z) \rangle$. Specifically, given a feature mapping ϕ , we define the corresponding **Kernel** to be

$$K(x, z) = \phi(x)^T \phi(z).$$

Now, let's talk about a slightly different view of kernels. Intuitively, (and there are things wrong with this intuition, but nevermind), if $\phi(x)$ and $\phi(z)$ are close together, then we might expect $K(x, z) = \phi(x)^T \phi(z)$ to be large. Conversely, if $\phi(x)$ and $\phi(z)$ are far apart—say nearly orthogonal to each other—then $K(x, z) = \phi(x)^T \phi(z)$ will be small. So, we can think of $K(x, z)$ as some measurement of how similar are $\phi(x)$ and $\phi(z)$, or of how similar are x and z .

Given this intuition, suppose that for some learning problem that you're working on, you've come up with some function $K(x, z)$ that you think might be a reasonable measure of how similar x and z are. For instance, perhaps you chose

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right).$$

This is a resonable measure of x and z 's similarity, and is close to 1 when x and z are close, and near 0 when x and z are far apart. Can we use this definition of K as the kernel in an SVM? In this particular example, the answer is yes. (This kernel is called the **Gaussian kernel**, and corresponds to an infinite dimensional feature mapping ϕ .) But more broadly, given some function K , how can we tell if it's a valid kernel; i.e., can we tell if there is some feature mapping ϕ so that $K(x, z) = \phi(x)^T \phi(z)$ for all x, z ?

Necessary and Sufficient Condition for a valid Kernel

Theorem (Mercer). Let $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^{(1)}, \dots, x^{(m)}\}$, ($m < \infty$), the corresponding kernel matrix is symmetric positive semi-definite.

Given a function K , apart from trying to find a feature mapping ϕ that corresponds to it, this theorem therefore gives another way of testing if it is a valid kernel.

- Work out details with the students on the blackboard.

Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ . Now, consider some finite set of m points (not necessarily the training set) $\{x^{(1)}, \dots, x^{(m)}\}$, and let a square, m -by- m matrix K be defined so that its (i, j) -entry is given by $K_{ij} = K(x^{(i)}, x^{(j)})$. This matrix is called the **Kernel matrix**. Note that we've overloaded the notation and used K to denote both the kernel function $K(x, z)$ and the kernel matrix K , due to their obvious close relationship.

K is positive semi-definite

Now, if K is a valid Kernel, then $K_{ij} = K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)}) = \phi(x^{(j)})^T \phi(x^{(i)}) = K(x^{(j)}, x^{(i)}) = K_{ji}$, and hence K must be symmetric. Moreover, letting $\phi_k(x)$ denote the k -th coordinate of the vector $\phi(x)$, we find that for any vector z , we have

$$\begin{aligned} z^T K z &= \sum_i \sum_j z_i K_{ij} z_j \\ &= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j \\ &= \sum_i \sum_j z_i \sum_k \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\ &= \sum_k \sum_i \sum_j z_i \phi_k(x^{(i)}) \phi_k(x^{(j)}) z_j \\ &= \sum_k \left(\sum_i z_i \phi_k(x^{(i)}) \right)^2 \\ &\geq 0. \end{aligned}$$

- More details on Support Vector Machine
- The following slides are read only

Model the problem into an optimization problem

$$\begin{aligned} \max_{\gamma, w, b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq \gamma, \quad i = 1, \dots, m \\ & \|w\| = 1. \end{aligned}$$

- *But this is not a convex problem!*
- *How to change it to a convex problem?*
- *Possible?*

Cleverly build $\|w\|$ into the cost and use scalar invariant smartly.

$$\begin{aligned} & \max_{\hat{\gamma}, w, b} \frac{\hat{\gamma}}{\|w\|} \\ \text{s.t. } & y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, \quad i = 1, \dots, m \end{aligned}$$

- Recall: we can add an arbitrary scaling constraint on w and b without changing anything. This is the key idea we are going to use cleverly.
- We will introduce the scaling constraint that the functional margin of w, b with respect to the training set must be 1.

$$\hat{\gamma} = 1.$$

Successfully transform the original optimization problem into a convex optimization problem.

$$\begin{aligned} \min_{\gamma, w, b} \quad & \frac{1}{2} \|w\|^2 \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \dots, m \end{aligned}$$

This is a convex optimization problem since the objective function is convex and the constraints are linear/affine.

The solution to this problem is called the *optimal margin classifier*.

This optimization problem can be solved using commercial **quadratic programming (QP)** code.

Lagrange Duality

Consider a general problem:

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.t.} \quad & h_i(w) = 0, \quad i = 1, \dots, l. \end{aligned}$$

Define **the Lagrangian**:

$$\mathcal{L}(w, \beta) = f(w) + \sum_{i=1}^l \beta_i h_i(w)$$

Here, the β_i 's are called the *Lagrange multipliers*.

Key steps involved

- Find the partial derivatives and set them to zero.

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0; \quad \frac{\partial \mathcal{L}}{\partial \beta_i} = 0,$$

- Solve for w and β .
- *Q: How to deal with constrained optimization problems in which there are inequality as well as equality constraint?*

Consider

$$\min_w \quad f(w)$$

$$\text{s.t.} \quad g_i(w) \leq 0, \quad i = 1, \dots, k$$

$$h_i(w) = 0, \quad i = 1, \dots, l.$$

This optimization problem is called
the primal optimization problem.

- Back up slides
- Read only

Generalized Lagrangian

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w).$$

Here, the α_i 's and β_i 's are the Lagrange multipliers.

Similar as before.

Key: How to get rid of the inequalities?

Consider:

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta).$$



Primal optimization

$$\theta_{\mathcal{P}}(w) = \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta).$$

Here, the “ \mathcal{P} ” subscript stands for “primal.” Let some w be given. If w violates any of the primal constraints (i.e., if either $g_i(w) > 0$ or $h_i(w) \neq 0$ for some i), then you should be able to verify that

$$\begin{aligned}\theta_{\mathcal{P}}(w) &= \max_{\alpha, \beta : \alpha_i \geq 0} f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w) \quad (1) \\ &= \infty.\end{aligned}$$

Conversely, if the constraints are indeed satisfied for a particular value of w , then $\theta_{\mathcal{P}}(w) = f(w)$. Hence,

$$\theta_{\mathcal{P}}(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{otherwise.} \end{cases}$$

We essentially be able to get rid of the inequalities.

Thus, $\theta_{\mathcal{P}}$ takes the same value as the objective in our problem for all values of w that satisfies the primal constraints, and is positive infinity if the constraints are violated. Hence, if we consider the minimization problem

$$\min_w \theta_{\mathcal{P}}(w) = \min_w \max_{\alpha, \beta : \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta),$$

we see that it is the same problem (i.e., and has the same solutions as) our original, primal problem. For later use, we also define the optimal value of the objective to be $p^* = \min_w \theta_{\mathcal{P}}(w)$; we call this the **value** of the primal problem.

The Dual Optimization Problem

Now, let's look at a slightly different problem. We define

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_w \mathcal{L}(w, \alpha, \beta).$$

Here, the “ \mathcal{D} ” subscript stands for “dual.” Note also that whereas in the definition of $\theta_{\mathcal{P}}$ we were optimizing (maximizing) with respect to α, β , here we are minimizing with respect to w .

We can now pose the **dual** optimization problem:

$$\max_{\alpha, \beta : \alpha_i \geq 0} \theta_{\mathcal{D}}(\alpha, \beta) = \max_{\alpha, \beta : \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta).$$

This is exactly the same as our primal problem shown above, except that the order of the “max” and the “min” are now exchanged. We also define the optimal value of the dual problem’s objective to be $d^* = \max_{\alpha, \beta : \alpha_i \geq 0} \theta_{\mathcal{D}}(w)$.

Relations between Max-Min and Min-Max Problems

How are the primal and the dual problems related? It can easily be shown that

$$d^* = \max_{\alpha, \beta: \alpha_i \geq 0} \min_w \mathcal{L}(w, \alpha, \beta) \leq \min_w \max_{\alpha, \beta: \alpha_i \geq 0} \mathcal{L}(w, \alpha, \beta) = p^*.$$

(You should convince yourself of this; this follows from the “max min” of a function always being less than or equal to the “min max.”) However, under certain conditions, we will have

*The proof is out of scope of
This course.*

$$d^* = p^*,$$

so that we can solve the dual problem in lieu of the primal problem. Let's see what these conditions are.

Q: When do the primal and dual problems have same solution?

Suppose f and the g_i 's are convex,³ and the h_i 's are affine.⁴ Suppose further that the constraints g_i are (strictly) feasible; this means that there exists some w so that $g_i(w) < 0$ for all i .

Under our above assumptions, there must exist w^*, α^*, β^* so that w^* is the solution to the primal problem, α^*, β^* are the solution to the dual problem, and moreover $p^* = d^* = \mathcal{L}(w^*, \alpha^*, \beta^*)$. Moreover, w^*, α^* and β^* satisfy the **Karush-Kuhn-Tucker (KKT) conditions**

³When f has a Hessian, then it is convex if and only if the Hessian is positive semi-definite. For instance, $f(w) = w^T w$ is convex; similarly, all linear (and affine) functions are also convex. (A function f can also be convex without being differentiable, but we won't need those more general definitions of convexity here.)

⁴I.e., there exists a_i, b_i , so that $h_i(w) = a_i^T w + b_i$. “Affine” means the same thing as linear, except that we also allow the extra intercept term b_i .

KKT Conditions

Karush-Kuhn-Tucker (KKT) conditions, which are as follows:

$$\frac{\partial}{\partial w_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, n \quad (3)$$

$$\frac{\partial}{\partial \beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, l \quad (4)$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \dots, k \quad (5)$$

$$g_i(w^*) \leq 0, \quad i = 1, \dots, k \quad (6)$$

$$\alpha^* \geq 0, \quad i = 1, \dots, k \quad (7)$$

Moreover, if some w^*, α^*, β^* satisfy the KKT conditions, then it is also a solution to the primal and dual problems.

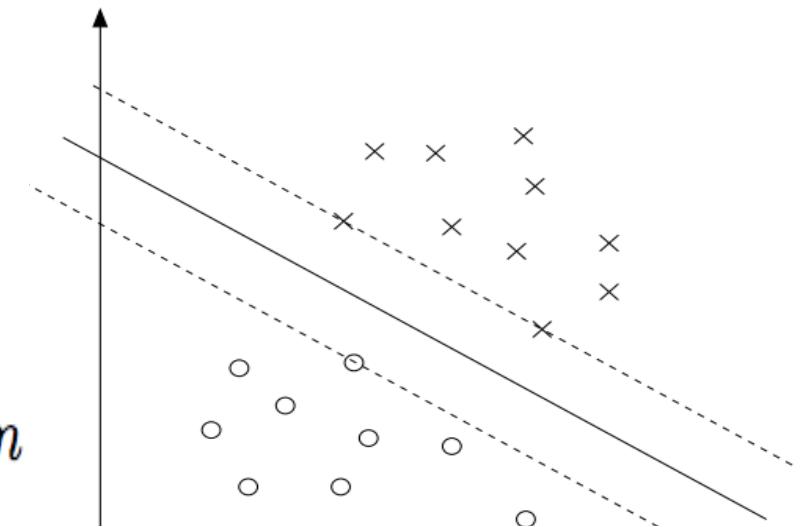
KKT dual complementarity condition

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \dots, k \quad (5)$$

We draw attention to Equation (5), which is called the **KKT dual complementarity** condition. Specifically, it implies that if $\alpha_i^* > 0$, then $g_i(w^*) = 0$. (I.e., the “ $g_i(w) \leq 0$ ” constraint is **active**, meaning it holds with equality rather than with inequality.) Later on, this will be key for showing that the SVM has only a small number of “support vectors”; the KKT dual complementarity condition will also give us our convergence test when we talk about the SMO algorithm.

Optimal margin classifiers--SVM

$$\begin{aligned} & \min_{\gamma, w, b} \frac{1}{2} \|w\|^2 \\ \text{s.t. } & y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \dots, m \end{aligned}$$



We can write the constraints as

$$g_i(w) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0.$$

We have one such constraint for each training example. Note that from the KKT dual complementarity condition, we will have $\alpha_i > 0$ only for the training examples that have functional margin exactly equal to one (i.e., the ones corresponding to constraints that hold with equality, $g_i(w) = 0$). Consider the figure below, in which a maximum margin separating hyperplane is shown by the solid line.

Back to SVM

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y^{(i)}(w^T x^{(i)} + b) - 1].$$

Note that there're only “ α_i ” but no “ β_i ” Lagrange multipliers, since the problem has only inequality constraints.

Let's find the dual form of the problem. To do so, we need to first minimize $\mathcal{L}(w, b, \alpha)$ with respect to w and b (for fixed α), to get θ_D , which we'll do by setting the derivatives of \mathcal{L} with respect to w and b to zero. We have:

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0$$

This implies that

$$w = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}. \quad (9)$$

As for the derivative with respect to b , we obtain

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y^{(i)} = 0. \quad (10)$$

If we take the definition of w in Equation (9) and plug that back into the Lagrangian (Equation 8), and simplify, we get

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^m \alpha_i y^{(i)}.$$

But from Equation (10), the last term must be zero, so we obtain

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}.$$

Dual Optimization Problem-SVM

Recall that we got to the equation above by minimizing \mathcal{L} with respect to w and b . Putting this together with the constraints $\alpha_i \geq 0$ (that we always had) and the constraint (10), we obtain the following dual optimization problem:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle.$$

$$\text{s.t. } \alpha_i \geq 0, \quad i = 1, \dots, m$$

$$\sum_{i=1}^m \alpha_i y^{(i)} = 0,$$



Key: All the x-data is in this inner product!

From this dual point view, we will see SVM can be generated by replacing the inner product by a Kernel!

By examining the dual form of the optimization problem, we gained significant insight into the structure of the problem, and were also able to write the entire algorithm in terms of only inner products between input feature vectors. In the next section, we will exploit this property to apply the kernels to our classification problem. The resulting algorithm, **support vector machines**, will be able to efficiently learn in very high dimensional spaces.

Exercise

You should also be able to verify that the conditions required for $p^* = d^*$ and the KKT conditions (Equations 3–7) to hold are indeed satisfied in our optimization problem. Hence, we can solve the dual in lieu of solving the primal problem. Specifically, in the dual problem above, we have a maximization problem in which the parameters are the α_i 's. We'll talk later about the specific algorithm that we're going to use to solve the dual problem, but if we are indeed able to solve it (i.e., find the α 's that maximize $W(\alpha)$ subject to the constraints), then we can use Equation (9) to go back and find the optimal w 's as a function of the α 's. Having found w^* , by considering the primal problem, it is also straightforward to find the optimal value for the intercept term b as

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}. \quad (11)$$

Cool thing about this dual method

Before moving on, let's also take a more careful look at Equation (9), which gives the optimal value of w in terms of (the optimal value of) α . Suppose we've fit our model's parameters to a training set, and now wish to make a prediction at a new point input x . We would then calculate $w^T x + b$, and predict $y = 1$ if and only if this quantity is bigger than zero. But using (9), this quantity can also be written:

$$w^T x + b = \left(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \right)^T x + b \quad (12)$$

$$= \sum_{i=1}^m \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \quad (13)$$

Hence, if we've found the α_i 's, in order to make a prediction, we have to calculate a quantity that depends only on the inner product between x and the points in the training set. Moreover, we saw earlier that the α_i 's will all be zero except for the support vectors. Thus, many of the terms in the sum above will be zero, and we really need to find only the inner products between x and the support vectors (of which there is often only a small number) in order calculate (13) and make our prediction.

The SMO algorithm (Read and Exercise)

$$\max_{\alpha} \quad W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \quad (17)$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m \quad (18)$$

$$\sum_{i=1}^m \alpha_i y^{(i)} = 0. \quad (19)$$

Let's say we have set of α_i 's that satisfy the constraints (18-19). Now, suppose we want to hold $\alpha_2, \dots, \alpha_m$ fixed, and take a coordinate ascent step and reoptimize the objective with respect to α_1 . Can we make any progress? The answer is no, because the constraint (19) ensures that

$$\alpha_1 y^{(1)} = - \sum_{i=2}^m \alpha_i y^{(i)}.$$

Or, by multiplying both sides by $y^{(1)}$, we equivalently have

$$\alpha_1 = -y^{(1)} \sum_{i=2}^m \alpha_i y^{(i)}.$$

(This step used the fact that $y^{(1)} \in \{-1, 1\}$, and hence $(y^{(1)})^2 = 1$.) Hence, α_1 is exactly determined by the other α_i 's, and if we were to hold $\alpha_2, \dots, \alpha_m$ fixed, then we can't make any change to α_1 without violating the constraint (19) in the optimization problem.

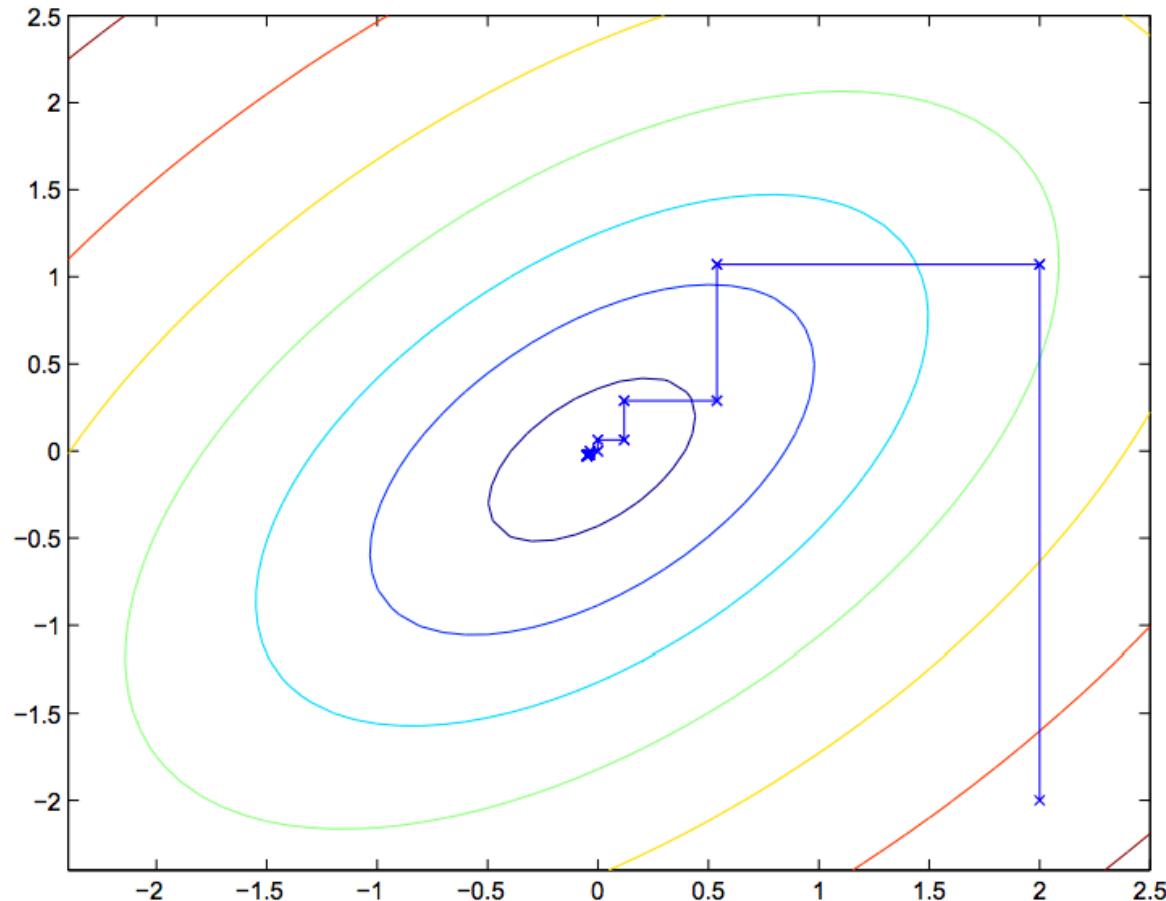
Thus, if we want to update some subject of the α_i 's, we must update at least two of them simultaneously in order to keep satisfying the constraints. This motivates the SMO algorithm, which simply does the following:

Repeat till convergence {

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Reoptimize $W(\alpha)$ with respect to α_i and α_j , while holding all the other α_k 's ($k \neq i, j$) fixed.

}

To test for convergence of this algorithm, we can check whether the KKT conditions (Equations 14-16) are satisfied to within some tol . Here, tol is the convergence tolerance parameter, and is typically set to around 0.01 to 0.001. (See the paper and pseudocode for details.)



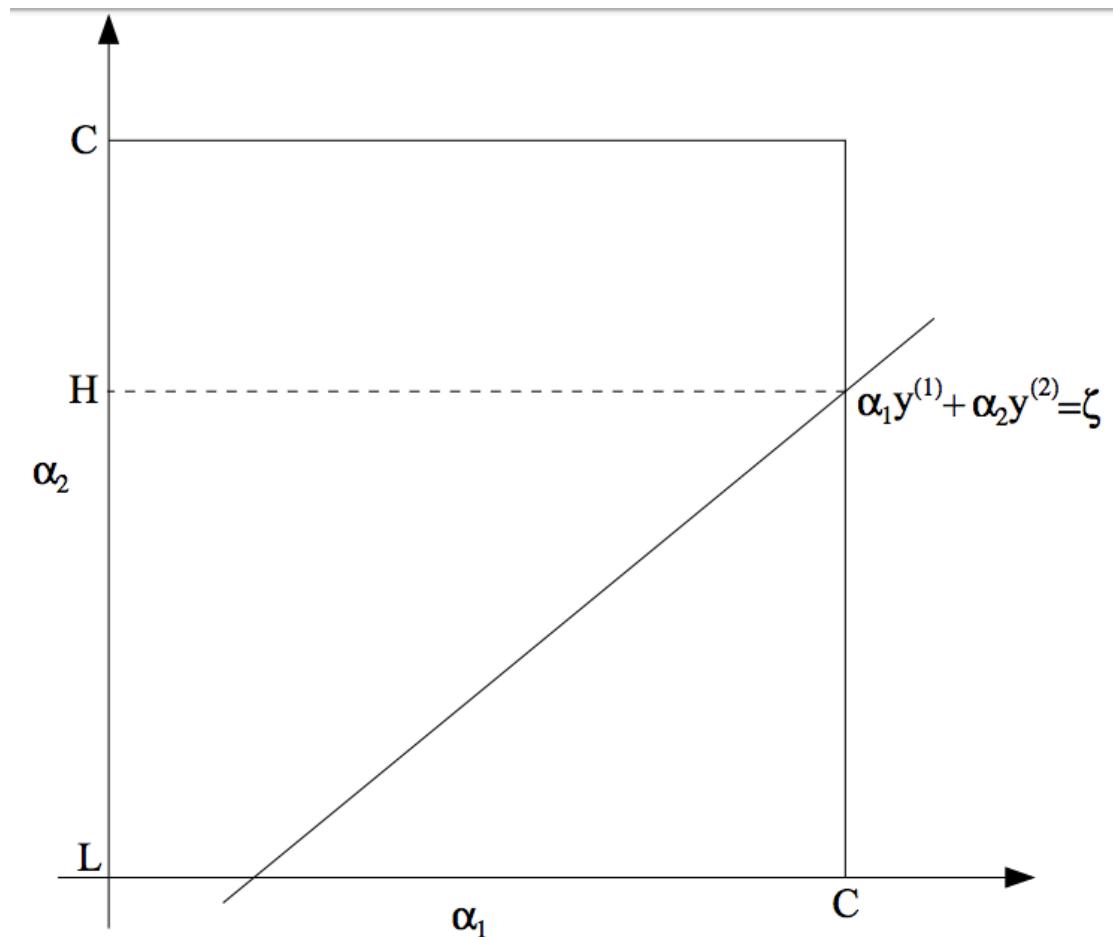
The key reason that SMO is an efficient algorithm is that the update to α_i , α_j can be computed very efficiently. Let's now briefly sketch the main ideas for deriving the efficient update.

Let's say we currently have some setting of the α_i 's that satisfy the constraints (18-19), and suppose we've decided to hold $\alpha_3, \dots, \alpha_m$ fixed, and want to reoptimize $W(\alpha_1, \alpha_2, \dots, \alpha_m)$ with respect to α_1 and α_2 (subject to the constraints). From (19), we require that

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = - \sum_{i=3}^m \alpha_i y^{(i)}.$$

Since the right hand side is fixed (as we've fixed $\alpha_3, \dots, \alpha_m$), we can just let it be denoted by some constant ζ :

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta. \tag{20}$$



From the constraints (18), we know that α_1 and α_2 must lie within the box $[0, C] \times [0, C]$ shown. Also plotted is the line $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta$, on which we know α_1 and α_2 must lie. Note also that, from these constraints, we know $L \leq \alpha_2 \leq H$; otherwise, (α_1, α_2) can't simultaneously satisfy both the box and the straight line constraint. In this example, $L = 0$. But depending on what the line $\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \zeta$ looks like, this won't always necessarily be the case; but more generally, there will be some lower-bound L and some upper-bound H on the permissible values for α_2 that will ensure that α_1, α_2 lie within the box $[0, C] \times [0, C]$.

Using Equation (20), we can also write α_1 as a function of α_2 :

$$\alpha_1 = (\zeta - \alpha_2 y^{(2)}) y^{(1)}.$$

(Check this derivation yourself; we again used the fact that $y^{(1)} \in \{-1, 1\}$ so that $(y^{(1)})^2 = 1$.) Hence, the objective $W(\alpha)$ can be written

$$W(\alpha_1, \alpha_2, \dots, \alpha_m) = W((\zeta - \alpha_2 y^{(2)}) y^{(1)}, \alpha_2, \dots, \alpha_m).$$

Treating $\alpha_3, \dots, \alpha_m$ as constants, you should be able to verify that this is just some quadratic function in α_2 . I.e., this can also be expressed in the form $a\alpha_2^2 + b\alpha_2 + c$ for some appropriate a , b , and c . If we ignore the “box” constraints (18) (or, equivalently, that $L \leq \alpha_2 \leq H$), then we can easily maximize this quadratic function by setting its derivative to zero and solving.

We'll let $\alpha_2^{new,unclipped}$ denote the resulting value of α_2 . You should also be able to convince yourself that if we had instead wanted to maximize W with respect to α_2 but subject to the box constraint, then we can find the resulting value optimal simply by taking $\alpha_2^{new,unclipped}$ and “clipping” it to lie in the

$[L, H]$ interval, to get

$$\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } L \leq \alpha_2^{new,unclipped} \leq H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$$

Finally, having found the α_2^{new} , we can use Equation (20) to go back and find the optimal value of α_1^{new} .

There're a couple more details that are quite easy but that we'll leave you to read about yourself in Platt's paper: One is the choice of the heuristics used to select the next α_i , α_j to update; the other is how to update b as the SMO algorithm is run.